2009年度 修士論文

ビジュアルプログラミング言語における
プログラム清書システムの構築

提出日：2010年2月1日
指導：筧 捷彦 教授

早稲田大学大学院基幹理工学研究科
情報理工学専攻

学籍番号：5108B013-4
稲垣拓海

目 次

第1章　序論

本論文の主題であるビジュアルプログラミング言語におけるプログラム清書システムについて述べる．著者がこのシステムを開発しようと考えた背景, 本研究の目的, そして論文全体の構成を述べていく．

1.1　研究の背景

近年，多くの人々がインターネットを通じて様々な web サービスを利用するようになった．広く利用されている web サービスの多くは，特別な知識なしに利用することができる．そのため誰でも容易にサービスを受けることができる．しかし，web サービスを発信するためには幅広い知識が必要とされる．web サービス利用者の中で web に関する知識があまりない人は少なくない．そのような人々でも web サービスを利用していく中で，新しい web サービスのアイディアが浮かぶことがあるはずだ．そんなアイディアを実現するために，初心者でも使いやすい開発環境が必要となる．そこで目をつけたのがビジュアルプログラミング言語を用いた開発環境だ．

初心者にとって一番の問題点はプログラムを作成する部分である．一般的なテキストベースのプログラミング言語は，構文規則などプログラムの記述方法を理解する必要があるため敷居が高い．一方，ビジュアルプログラミング言語ならば，直感的な操作ができるため初心者でも利用しやすい．また，表やグラフを使用することで，テキストでは表現しにくい複雑な構造や関係を簡単に表現することができる．そのためプログラム全体を把握しやすいという大きな利点がある．初心者が使いやすい開発環境ではビジュアルプログラミング言語が最適であると考えた．

しかし，それでもまだ初心者にとってプログラミングの敷居は高い．プログラムが大きくなると，ビジュアルプログラミング言語でも全体を理解するのが難しくなる．そこで，プログラムを自動できれいに書き直すシステムが必要となると考えた．ビジュアルプログラミング言語に清書システムを実装することで，初心者でも使いやすい web サービス開発環境を提供することができるはずである．

1.2 研究の目的

ビジュアルプログラミング言語におけるプログラムの可読性の向上に特化したプログラム清書システムの開発を行う．本研究の目的は，複雑なプログラムでも読みやすい形に清書することで，誰もが理解しやすいプログラムを作成することである．ビジュアルプログラミング言語は視覚的に表現することができるため，プログラミング初心者でも読みやすいという特徴がある．しかし，プログラムが複雑になるとゴチャゴチャして非常に読みにくいものになってしまう．そんなときに，最も可読性が高い形に自動で書き換えるシステムがあれば便利だと考えた．誰もが読みやすいと思えるプログラムを定義して，それを実際にビジュアルプログラム言語に導入して評価することが今回の研究内容である．

1.3 論文の構成

本論分は全 5 章で構成されている．

第 1 章では，研究の背景, 研究の目的, そして論文の構成を述べた．

第 2 章では，清書システムを構築するための構想を述べる．

第 3 章では，第 2 章で考案した構想を，実際にビジュアルプログラミング言語へ実装するための設計ついて述べる．

第 4 章では，本論文で提示した清書システムが有用なものであるかの検証について述べる．

第 5 章では，論文全体のまとめを行う．

第2章　構想

ビジュアルプログラミング言語におけるプログラム清書システムを構築するための構想を述べる．プログラムを清書するには，視覚的にきれいで読みやすいという点と，内容がまとめられていて理解しやすいという点の両方を満たす必要がある．視覚的な観点からは，ビジュアルプログラミング言語をグラフと捉えて，美しいグラフとはどのようなものか考察する．プログラムの内容を整理するという観点からは，他のプログラミング言語に対する清書方法を調査して，それを基にビジュアルプログラミング言語でのまとめ方を考察する．

2.1　美しいグラフ

ビジュアルプログラミング言語は，要素をグラフィカルに操作することによってプログラムを作成するものである．オブジェクトを線等で繋いでいく様はいわゆるグラフそのものと言うことができる．

そこで，一般的に美しいグラフとはどのようなものかを調査した．美しいグラフの具体的な形も示すことできれいで読みやすいプログラムの形も見えてくる．美しいグラフになるようにプログラムを整理することで，清書と呼べるような可読性の高いプログラムとなるはずである．この節では美しいグラフとはどのようなものであるかを定義し，清書システムに取り入れる準備をしたい．

有向グラフ

本論文において，グラフというのは全て有向グラフのことを表す．ビジュアルプログラミング言語では，オブジェクト間には何らかの関係性があるため，それを繋ぐ線には必ず方向がある．つまり，ビジュアルプログラミング言語は有向グラフであると言える．有向グラフにおける美しいグラフを定義し，それを基に清書システムの構想を行う．

2.1.1　美しいグラフの基準

ビジュアルプログラミング言語を清書するにあたり，見た目の美しさを判断するための明確な基準を決める必要がある．美しさの感覚など人によって様々で，簡

単に決めることなどできない．しかし，大多数の人が美しいグラフと感じられるようなポイントはいくつか存在する．それらを組み合わせたグラフならば，それは美しいグラフと言えるはずである．そこで参考文献 [5], [6] などを参考に，明らかに美しいグラフであると言えるような基準をまとめた．どれもシンプルでわかりやすく，多くの人が美しいと感じられるポイントであるはずだ．

- 交差が少ない

  交差数は極力少なくする．交差が全くない，いわゆる平面グラフとなることが望ましい．

- 曲がりが少ない

  グラフ上の線の屈折回数は極力少なくする．

- 一方向に進む

  上から下や左から右のように決まった方向に進み，逆方向に戻らない．

- 等間隔に配置

  グラフ上の要素は等間隔に配置する．

- 対称的に配置

  グラフ上の要素が一部に密集しないように，バランスよくできるだけ対称になるように配置する．

これらの条件を満たしたグラフは，見た目がきれいで内容も理解しやすい，いわゆる美しいグラフといえる．

2.1.2 具体例による検証

2.1.1 で定めた美しいグラフの基準の検証を行う．それぞれの項目の具体例を示し，美しいグラフに必要な条件となる理由を示す．

交差が少ない

交差があるグラフの例を図 2.1 に示す．線が重なった交差点を挟んで，それぞれ対応する線がどれなのかが一見わからなくなる．そのため線の繋がった要素同士の対応関係もわかりにくくなってしまう．よって交差は少ないほうが良いと言える．交差をなくしたグラフの例を図 2.2 に示す．

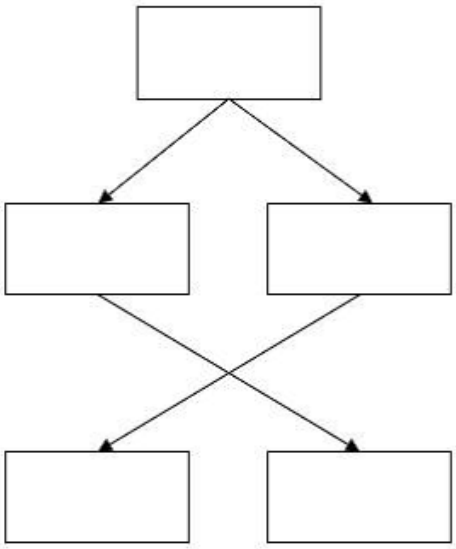

図 2.1: 交差あり

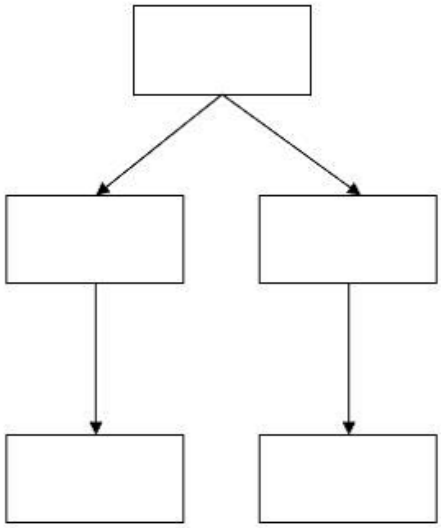

図 2.2: 交差なし

曲がりが少ない

曲がりがあるグラフの例を図 2.3 に示す．線の繋がった要素同士が一直線であれば対応先が一目ですぐにわかる．しかし，何度も屈折が起こった線は対応先がわかりにくくなる。よって曲がりは少ないほうが良いと言える．曲がりをなくしたグラフの例を図 2.4 に示す.

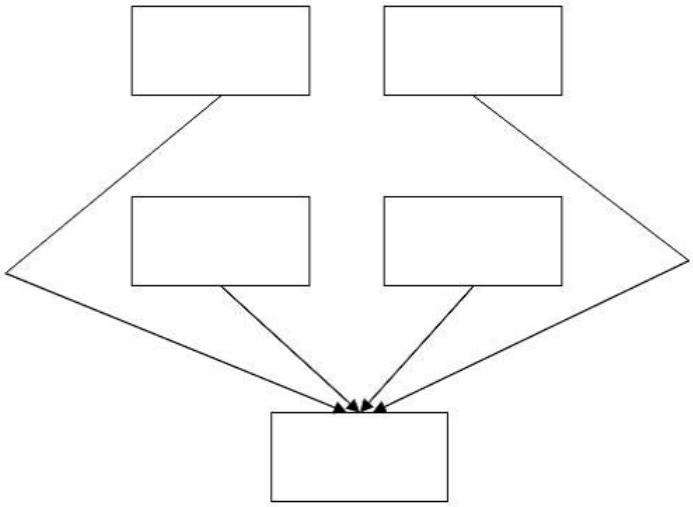

図 2.3: 曲がりあり

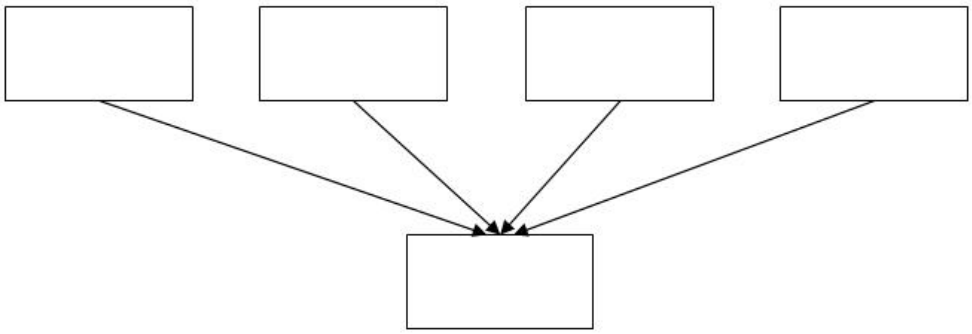

図 2.4: 曲がりなし

一方向に進む

進む方向が一方向でないグラフの例を図2.5に示す．一番下のレベルにある要素がどれであるか非常にわかりにくい．進む方向を一方向にしたグラフの例を図2.6に示す．線の流れが上から下に進んでいくので，全体の流れがわかりやすくなっている．よって一方向に進むグラフのほうが良いと言える．

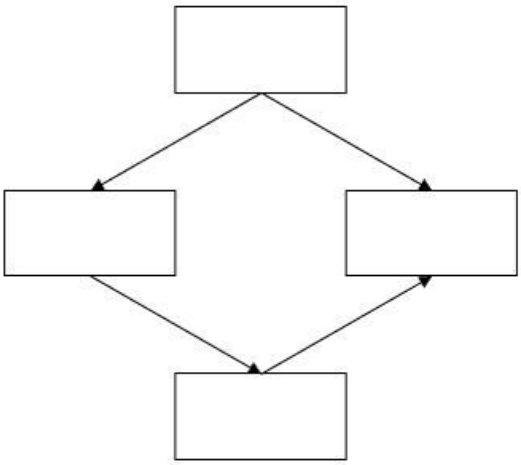

図 2.5: 一方向でない

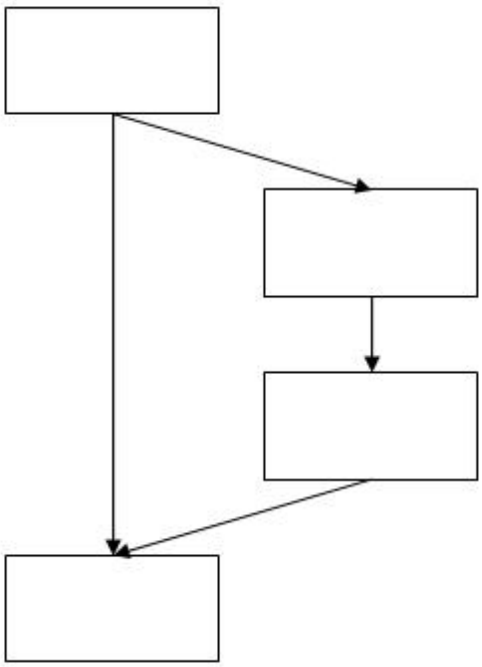

図 2.6: 一方向である

等間隔に配置

等間隔に配置されていないグラフの例を図 2.7 に示す．線の長さに差があり明らかにきれいではない．一方，等間隔に配置したグラフの例を図 2.8 に示す．線の長さが統一されていて見栄えが良くなる．よって等間隔に配置したグラフのほうが良いと言える．

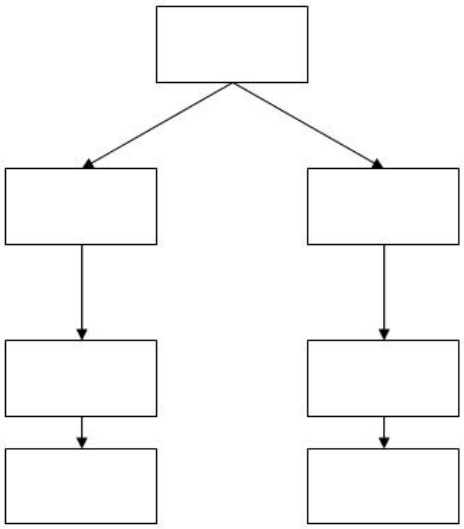

図 2.7: 等間隔でない

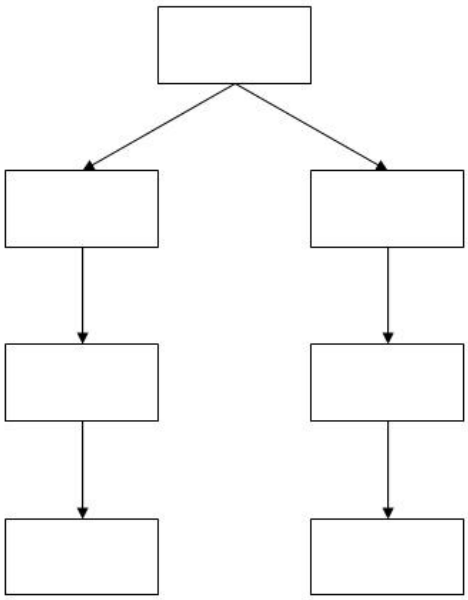

図 2.8: 等間隔である

対称的に配置

対称的に配置されていないグラフの例を図 2.9 に示す．繋がっている線の長さに差があり，左右で要素の数も異なっているためきれいでない．一方，対称的に配置したグラフの例を図 2.10 に示す．線の長さが等しく，左右の要素数も同一であるため見栄えが良い．よって対称的に配置したグラフのほうが良いと言える．

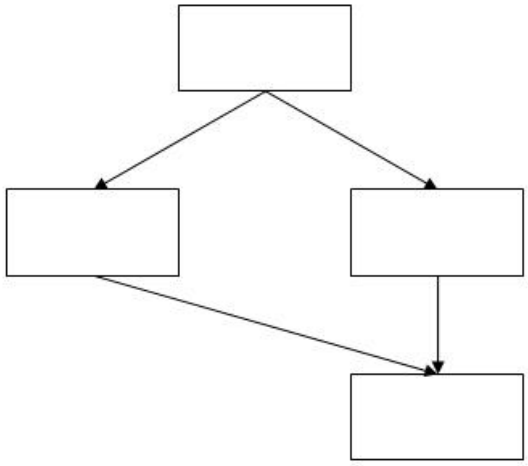

図 2.9: 対称的でない

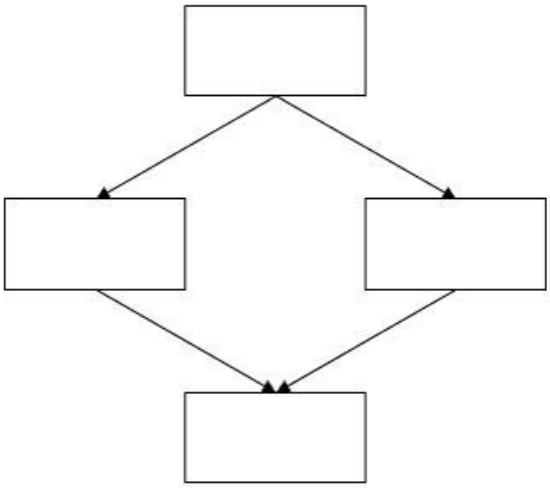

図 2.10: 対称的である

2.2　他のプログラミング言語に対する清書方法

ビジュアルプログラミング言語ではなく，一般的なテキストを用いたプログラミング言語での清書方法について調査した．ビジュアル面での違いは大きいが，プログラム要素のまとめ方などを参考にする．

2.2.1　構文解析

他のプログラミング言語に対する清書方法について調べてみたところ，可読性の高いプログラムを作るには構文解析をすることが重要だとわかった．与えられたプログラムを構文木の形にすることでプログラム全体の構造がわかる．その中で同じレベルにあるもの同士をまとめて書いていくことによって，可読性の高いプログラムを作成することができるというわけである．

2.2.2　ビジュアルプログラミング言語への適用

プログラムの全体的な形は，木構造を参考にして整理を行う。同じオブジェクトに繋がっているもの同士を同じレベルに配置していき，全体が木構造のような形になるように書き換えていく．ただし，ビジュアルプログラミング言語では輪になる部分ができるなど，厳密には木構造そのもので表すことはできない．木構造に近い形を目指してまとめていく．

木構造に倣って整理を行う理由は，他のプログラミング言語が構文木を用いてプログラムの清書を行っていることを参考にした．構文木の形にすることで，プログラム全体の構造がわかる．そして，同じレベルのもの同士をまとめることで可読性の高いプログラムにすることができる．ビジュアルプログラミング言語でも木構造にすることで，全体の構成が分かりやすい形に整理することができるはずである．

2.3　まとめ

この章で示した，ビジュアルプログラミング言語を清書するために必要なポイントをまとめ，プログラム清書システムを構築する方法の提案を行う．

プログラム全体としては，木構造に近い形で整理を行う．同じレベルの要素を同じ位置にまとめて並べていく．このプログラムをきれいに書き直すために，美しいグラフの基準を満たす形に仕上げていく．交差が少ない，曲がりが少ない，一方向に進む，等間隔に配置，対称的に配置という 5 つのポイントを抑えていれば清書されたプログラムであると言える．

次の第 3 章では，この章で構想したプログラム清書システムを，実際にビジュアルプログラミング言語に実装する．

第3章　設計

第 2 章で構想したプログラム清書システムを，実際にビジュアルプログラミング言語に実装する．本研究では，著者が研究に携わってきたビジュアルプログラミング言語「ゆば」に清書システムを構築していく．ゆばについて，詳しくは参考文献 [1], [2] を参照されたい．

3.1　ビジュアルプログラミング言語「ゆば」

ビジュアルプログラミング言語「ゆば」について詳しく述べる．

3.1.1　ゆばとは

ゆばとは，早稲田大学筧研究室のメンバーによって開発が行われている開発環境「YubaEE」および「MYuba」において使用されているビジュアルプログラミング言語のことである．ゆばの特徴は次のようになっている．

初心者向けプログラミング言語

ゆばは，初心者でも簡単にプログラミングを行うことができるように設計されている．ゆばはブラウザのみを使ってプログラミングを行うことができる．開発前の準備段階として，開発環境のインストールや環境設定や，web サービス公開用サーバの準備といった，開発以外の手間を省くことができる．

プログラムの作成は，図形を描画することによって行われる．このとき一部の操作を除き，マウス操作のみでプログラミングを行うことができる．煩雑はキーボード操作によるプログラミングを極力減らし，わかりやすく効率のいいプログラミングを行うことができる．よって，プログラミング言語を知らなくても，誰でも直感的にプログラミングを行えるのである．

データフロー型ビジュアルプログラミング言語

データフロー型ビジュアルプログラミング言語とは，演算命令の間で受け渡しされるデータの授受関係を図形表現したものである．図形要素はそれぞれが演算命令である．演算命令にはその入力と出力である端子があり，命令同士の出力端子・入力端子を線でつないでプログラムとする．データフロー型ビジュアルプログラミング言語においてプログラマは，命令の実行順序を明示的には指定しない．順序を記述しないことは，並列的な処理を自然に記述することを可能にする．

3.1.2 ゆばでのプログラミング

ゆばでは，図形を組み合わせることでプログラムの作成を行う．プログラムを構成する図形としてキャンバス, 端子, パネル, パイプの４種類が存在する．プログラムは，キャンバス上にパネルとパイプを配置して計算処理を記述したものである．

キャンバス

図 3.1 に示すグリッドの入った矩形をキャンバスと呼ぶ．プログラムの作成はキャンバス上に端子，パネル，パイプといった図形を組み合わせて配置することで行う．

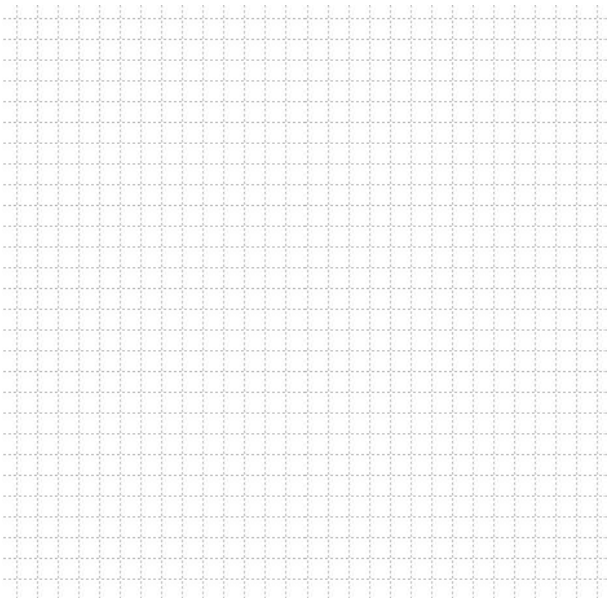

図 3.1: キャンバス

端子

図 3.2 から図 3.5 に示す図形を端子と呼ぶ．端子には入力端子, 結果端子, 出力端子，引数端子の 4 種類がある．入力端子と結果端子は角が丸くなっており，出力端子と引数端子は角が四角くなっている．入力端子と出力端子はパネルに付属するが，引数端子と結果端子はパネルに付属しない．

図 3.2: 入力端子

図 3.3: 結果端子

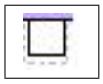

図 3.4: 出力端子

図 3.5: 引数端子

パネル

図 3.6 に示す図形をパネルと呼ぶ．パネルには 1 つ以上の入力端子もしくは出力端子が付いている．パネル名はパネルの中央に表示される．パネルの種類によっては，入力端子や出力端子の個数を変えられるものも存在する．

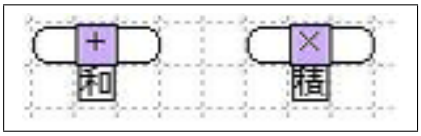

図 3.6: パネル

パイプ

図 3.7 に示す帯状の図形をパイプと呼ぶ．パイプは入力端子と出力端子を繋ぎ，データを運ぶ道の役割を果たす．パイプはクリエイタが端子を 2 つ選択することで自動的に最短経路で配置される．ちなみに，同じパネルの入力端子と出力端子を繋ぐことはできない．まだデータを出力する端子同士（出力端子，引数端子）やデータが入力される端子同士（入力端子，結果端子）を繋ぐことはできない．

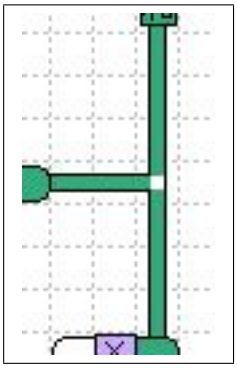

図 3.7: パイプ

引数端子と結果端子

引数端子と結果端子はキャンバス上に配置することができる．引数端子はキャンバスの上辺に沿い，マス目を各 1 個覆うように重ねて 0 個以上配置できる．結果端子はキャンバスの下辺に沿い，マス目を各 1 個覆うように重ねて 1 個以上配置できる．結果端子は 1 つ以上配置しなければならない．同じマス目を複数の端子やパネルが覆ってはいけない．

プログラムの部品化

プログラムは，部品化を行うとパネルとしてプログラムに配置することができるようになる．

プログラムをパネルとして使用する場合，そのプログラムが持つ入力端子と出力端子が，パネルの引数端子と結果端子に対応する．ページをパネルとして使用する場合，そのページが持つ引数がパネルの引数端子に，ページが出力する HTML 文字列がパネルの結果端子に対応する．そして，部品化されたパネルは他のパネルと同様に扱うことができる．

プログラム内に自分自身をパネルとして配置することで再帰呼出しを記述することができる．これによってゆばでは繰返し構造を作成する．

3.1.3　図形の動作

ゆばでのプログラミングにおける各構成図形の動作について説明する．

キャンバスの動作

キャンバスは端子，パネル，パイプなどの土台となるものである．それ自身が動作を行うことはない．

端子の動作

端子は１つだけデータを保持することができる．端子はプログラムの実行中に，パネルやパイプにデータを受け渡す役割を果たす．

端子はデータを要求されたときに動作を行う．端子は，初めてデータを要求されたときにはまだデータを持っていない．入力端子の場合，端子は自分が繋がっているパイプにデータを要求する．出力端子の場合，端子は自分が所属するパネルにデータを要求する．

逆に，端子はデータを要求されるものでもある．入力端子がデータを要求される相手は，自分の所属するパネルである．出力端子がデータを要求される相手は，自分を繋ぐパイプの先にある入力端子（結果端子）である．

データを要求された端子は，データを持っているならそのデータを返す．例えば，出力端子が自分を繋ぐパイプの先にある入力端子からデータを要求された場合，自分が所属するパネルにデータを要求し，そのデータを入力端子に返す形になる．

パネルの動作

パネルは，パネルに所属する出力端子にデータを要求されたときに動作を行う．パネルはデータを要求されると，パネルの種類ごとに定められた演算を行う．そしてその結果を出力端子に帰す．演算の際，パネルは必要なデータを所属する入力端子に要求する．

パイプの動作

パイプは入力端子によってデータを要求されたときに動作を行う．データを要求された入力端子に繋がっている先の出力端子にデータを要求し，返ってきたデータを入力端子に渡す．

プログラム作成画面

ゆばでのプログラミングは図 3.8 に示すプログラム作成画面上で行う．ゆばでは，パネルやパイプといった図形を繋ぎ合わせてプログラムの作成を行う．

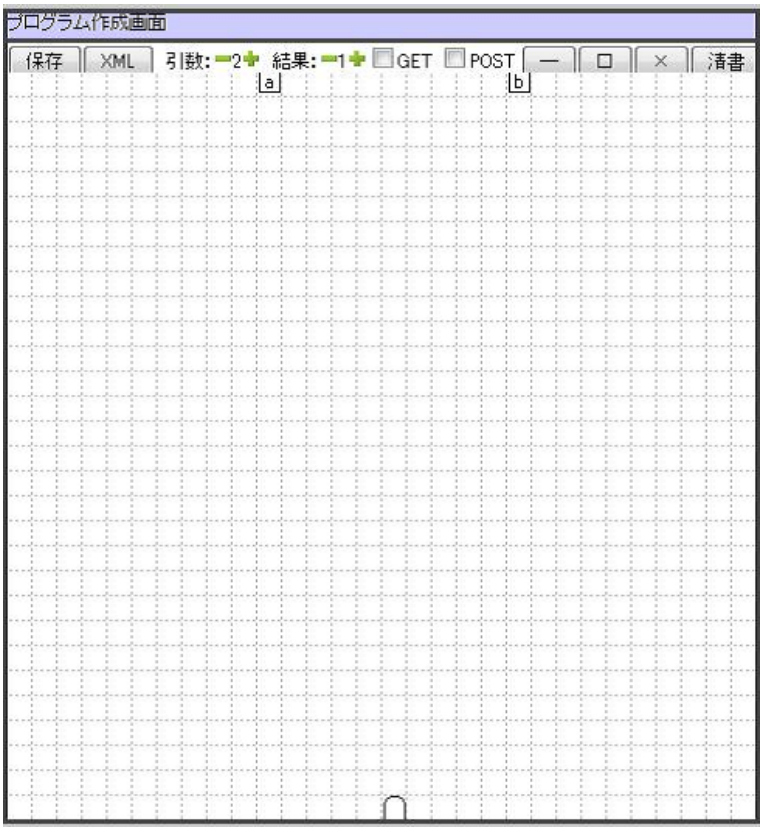

図 3.8: プログラム作成画面

パネルの配置

始めに，パネルをキャンバス上に配置する．部品領域に並んでいるパネルの中からキャンバス上に配置したいパネルをクリックする．部品領域はプログラム作成領域の右側にあり，使用可能なパネルを並べて配置してあるスペースのことである．

パネルをクリックするとパネルの背景が青色に変わる．これを「選択状態」という（図 3.9）.（選択状態はもう一度パネルをクリックすることで解除できる.）選択状態のときにキャンバス上でマウスを動かすと，マウスの動きに合わせたパネルの影が表示される（図 3.10）．この影ができている状態でクリックを行うと，影の場所にパネルが配置される．

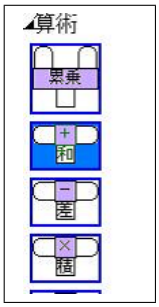

図 3.9: パネル選択状態

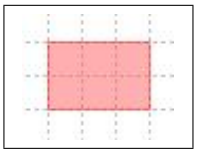

図 3.10: 選択パネルの影

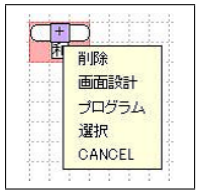

図 3.11: パネルの削除

パネルはすでに配置されているパネルやパイプの重なる場所に配置することはできない．パネルを配置できない場所では，パネルの影が消えるようになっている．

すでに配置されたパネルを右クリックするとポップアップメニューが表示される．その中の「削除」をクリックするとそのパネルがキャンバス上から削除される．(図 3.11) パネルが削除される際は，そのパネルに繋がったすべてのパイプも同時に削除される．

パイプの配置

次に，パイプを配置する．パイプはキャンバス上にある 2 つの端子を選択することで配置できる．

キャンバス上にあるパイプの 1 つをクリックすると，クリックした端子が「選択状態」になる（図 3.12）．その後もう 1 つの端子をクリックすることで，選択された 2 つの端子をパイプで繋ぐことができる．

一つのパイプを複数の入力端子に繋ぎたい場合はパイプの分岐を行う．パイプの分岐を行うには，すでに配置されたパイプを右クリックすると表示されるポップアップメニューから「分割」を選択する (図 3.13)．そうすると，パイプ上に分岐点が出現する (図 3.14)．その分岐点をクリックしてから入力端子をクリックすると，新たに同じパイプを配管することができる (図 3.15)．

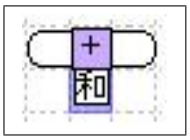

図 3.12: 端子選択状態

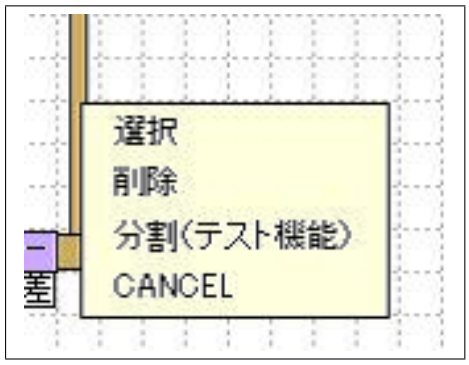

図 3.13: パイプのメニュー

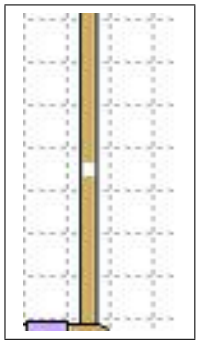

図 3.14: パイプの分岐点

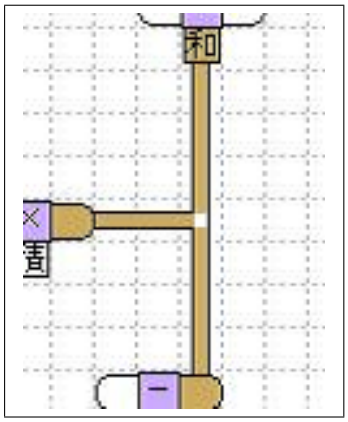

図 3.15: 分岐点からのパイプ

プログラムの操作

プログラム編集画面上にある「保存」ボタンをクリックすると，プログラムの保存をすることができる．プログラムの保存が完了すると，プログラム実行画面でそのプログラムを実行することができる．

プログラム編集画面上にある「□」ボタンをクリックすると，プログラムの最大化をすることができる．プログラムの最大化をすると，プログラムのサイズが作業領域全体に広がる．最大化の状態でプログラム編集画面上にある「◇」ボタンをクリックすると，プログラムは元のサイズに戻る．

プログラム編集画面上にある「−」ボタンをクリックすると，プログラムの最小化をすることができる．最小化されたプログラムは作業領域上から消え，タスクバーにあるプログラム名の表示色が変化する．最小化の状態でタスクバーにあるプログラム名をクリックすると，プログラムは元の作業領域に戻る．

プログラム編集画面上にある「×」ボタンをクリックすると，プログラム編集画面を閉じることができる．

図 3.16: プログラム操作メニュー

3.1.4 プログラムの作成例

ゆばでのプログラムの作成例として 3 つの数を入力としてその平均を返すプログラムを作成する．使用するパネルは図 3.17, 図 3.18, 図 3.19 に示す 3 つである．これらのパネルはそれぞれ， 入力端子に入力されたデータの和を出力端子に出力する「和」パネル， 左入力端子に入力されたデータを右入力端子に入力されたデータで割りその商を出力端子に出力する「比」パネル， パネル中央に表示されたテキストボックスに入力された値を出力端子に出力する「定数」パネルである． これらを図 3.20 のように組み合わせると 3 つの数の平均を計算するプログラムが作れる．

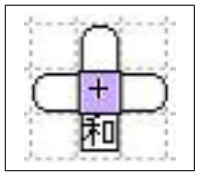

図 3.17: 和パネル

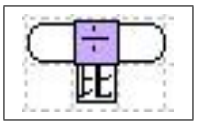

図 3.18: 比パネル

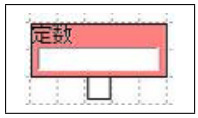

図 3.19: 定数パネル

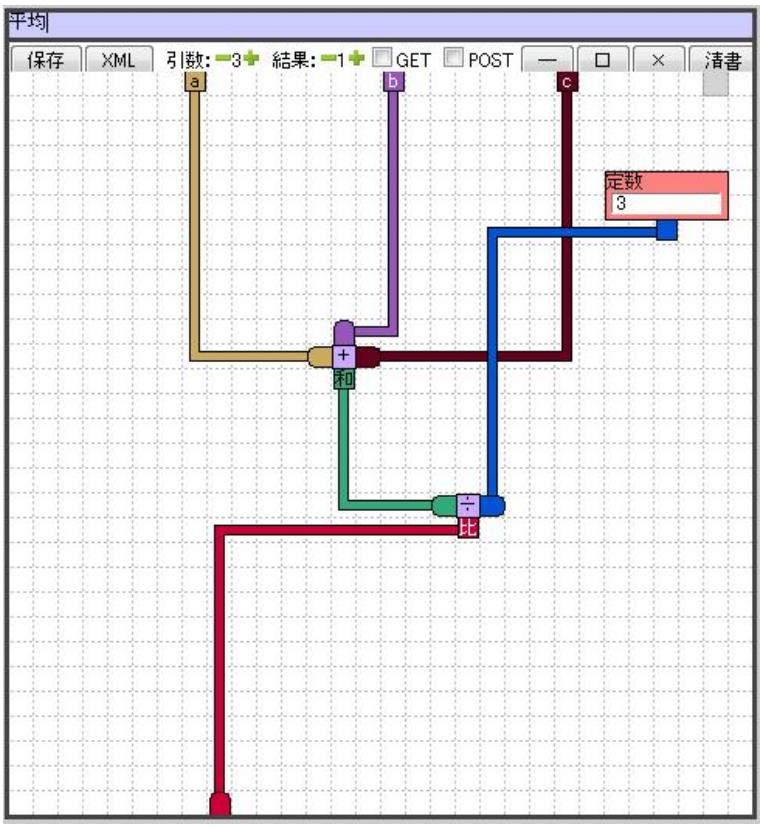

図 3.20: 3 つの引数の平均値を出力するプログラム

3.1.5　プログラムの実行例

ゆばで作成したプログラムは，ブラウザを通してプログラム実行画面にアクセスすることで実行することができる．この画面ではプログラムを指定することで，そのプログラムに対応する引数がテキストボックスの形式で表示される．3.1.4 で例示したプログラムを選択すると，a,b,c と 3 つのテキストボックスが表示される（図 3.21）．

テキストボックスに対して任意のデータを入力して送信ボタンをクリックすると画面繊維が行われ，遷移先の画面上に実行結果が表示される．例示したプログラムでは，a=300,b=30,c=3 とした．送信ボタンをクリックするとプログラムに 3 つの引数が渡され，”(300+30+3)/3”の結果である”111”がデータとして返ってくる．よって，遷移先画面には「111」と表示されることとなる (図 3.22).

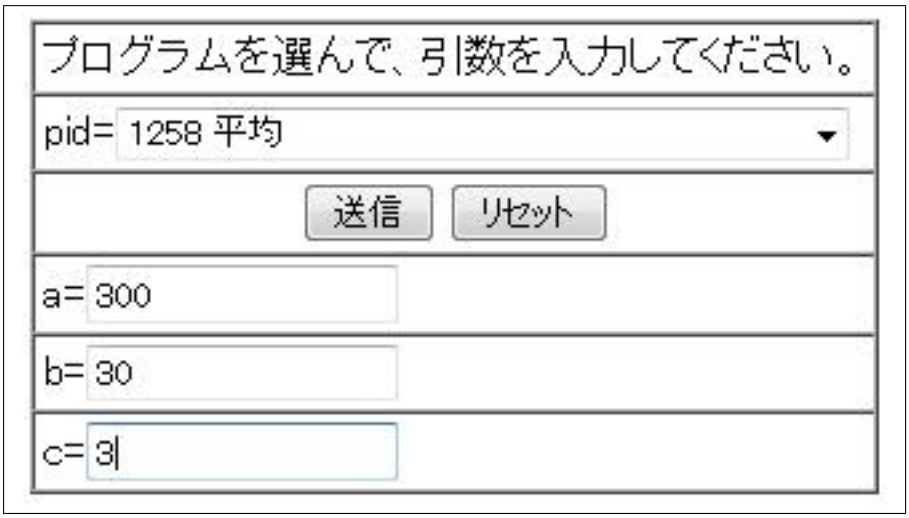

図 3.21: プログラム実行画面

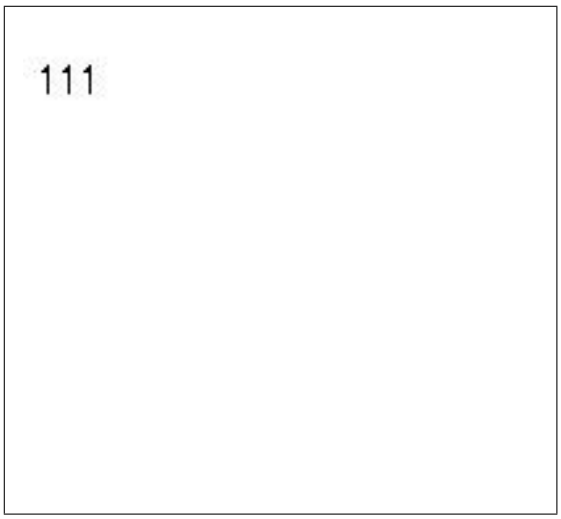

図 3.22: プログラム実行結果

3.2 アルゴリズム

ビジュアルプログラミング言語「ゆば」上に構築するプログラム清書システムのアルゴリズムを述べる．

3.2.1 全体の構造

プログラムの清書を行うためには木構造に整理することが望ましい．しかし，ゆばでは完全に木構造にすることは困難である．なぜなら，一つの出力端子から複数の入力端子への配管が行われた場合や，複数の出力をもつパネルを使用する場合に輪ができてしまうからだ．木では決して輪ができることはない．多くの場面で輪が発生するゆばにおいて，木そのもので表現することは難しいと言える．輪ができた場合はそれらをまとめて部品化することで対処できないことはない．しかし，最初の入力が複数の配管を行うとき，多くの場合で最終的にパネル一つの形に整理されてしまう．これではとても清書とは呼べない．

そこで本研究では，完全に木の形にはすることにはこだわらず，プログラムの清書に必要な部分だけを利用することにする．同じレベルのものを並べるという木構造に沿った整理の仕方は採用する．そのとき，プログラム中に輪ができてもかまわないものとする．

3.2.2 仕様

第 2 章で決めた美しいグラフの基準に沿ってゆばでの形式仕様を決める．この仕様に沿った形になるように，パネルの並べ替えやパネル自体の左右反転などを

行うようにする．

美しいグラフにするために，ゆばのプログラムをどのように書き換えていけば良いのかを次に示す．第 2 章で示したグラフ書き換えの例を実際にゆばで表現した．ゆばでは斜めにパイプを配管することはできないので表現が異なってくる部分はあるが，意味内容としては同様のものとなっている．

交差が少ない

交差が発生したパイプは，繋がっているパネルの配置を変えるまたはパネルの左右を反転させることによって，交差がなくなる場合が存在するかを調査する．これを繰り返してプログラム全体で最も交差が少ないパターンを決定する．

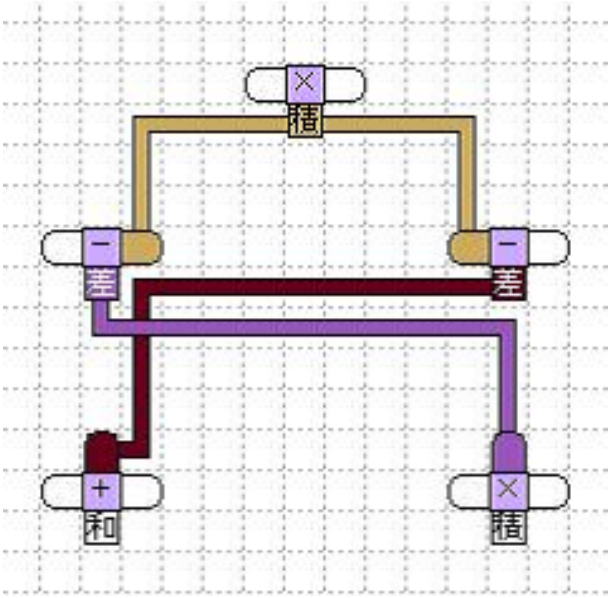

図 3.23: 交差あり

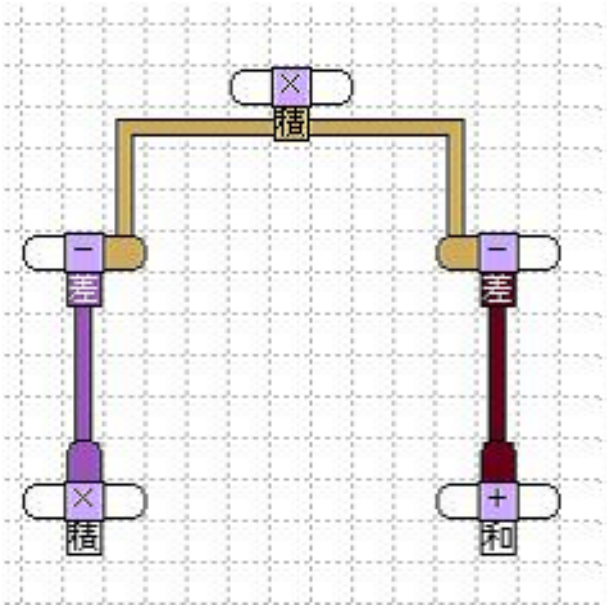

図 3.24: 交差なし

曲がりが少ない

曲がりが発生したパイプは，繋がっているパネルの配置を変えるまたはパイプを配管する順番を変えることによって，曲がった回数が少なくなる場合が存在するかを調査する．これを繰り返してプログラム全体で最も曲がりが少ないパターンを決定する．

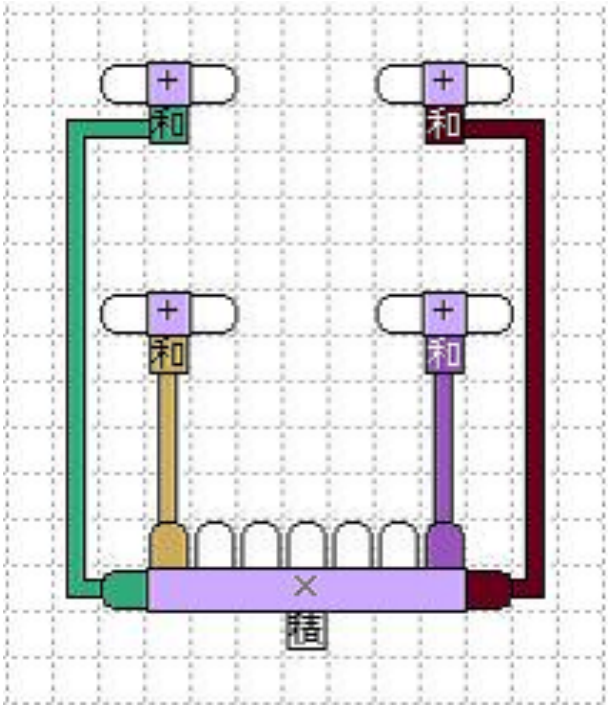

図 3.25: 曲がりが多い

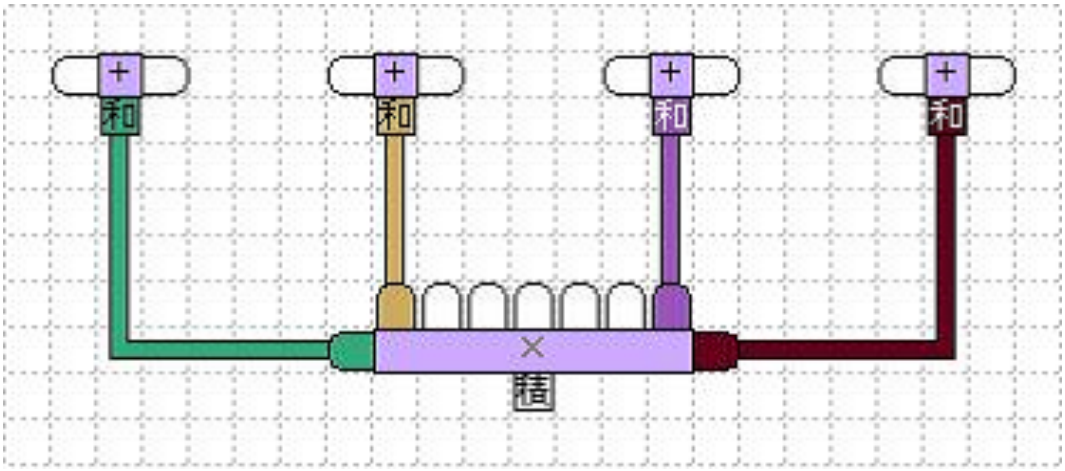

図 3.26: 曲がりが少ない

一方向に進む

キャンバス上で，下側にあるパネルから上側にあるパネルへの配管は絶対に行わないようにする．これにより必ず一方向に進むプログラムにすることができる．プログラムが縦に長くなりキャンバスに収まらなくなるときは，キャンバスサイズの変更を自動で行う．

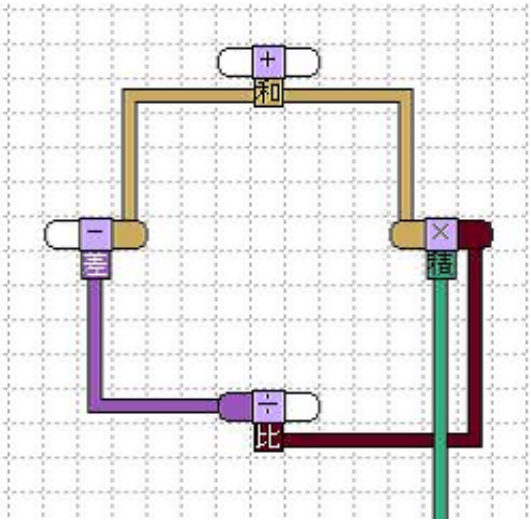

図 3.27: 一方向でない

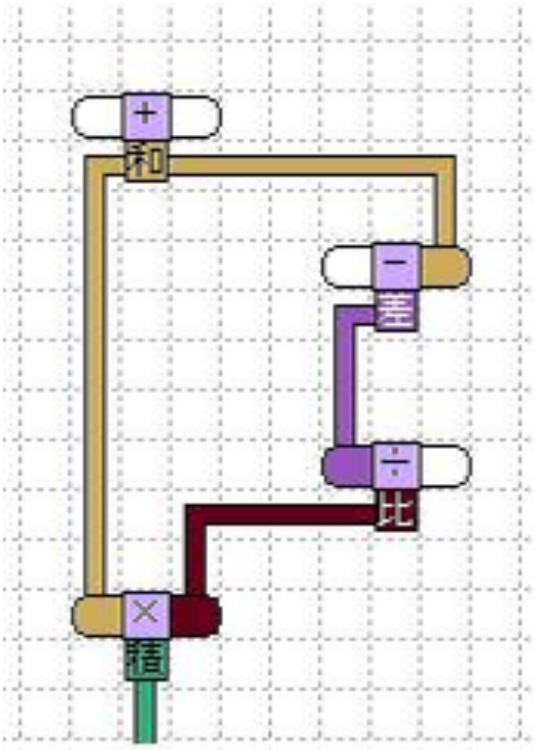

図 3.28: 一方向である

等間隔に配置

パネル間の距離は常に一定になるようにする．このとき，一マス分は必ずスペースを空けるようにする．パネル間が密接してしまうとゴチャゴチャして圧迫感のあるプログラムになってしまうからである．プログラムが縦に長くなりキャンバスに収まらなくなるときは，キャンバスサイズの変更を自動で行う．

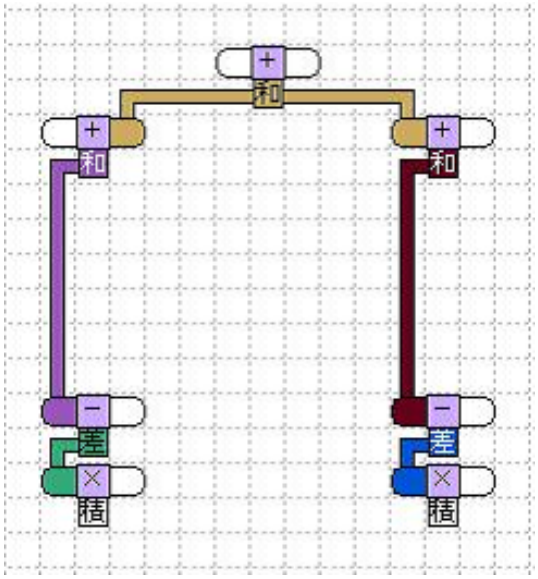

図 3.29: 等間隔でない

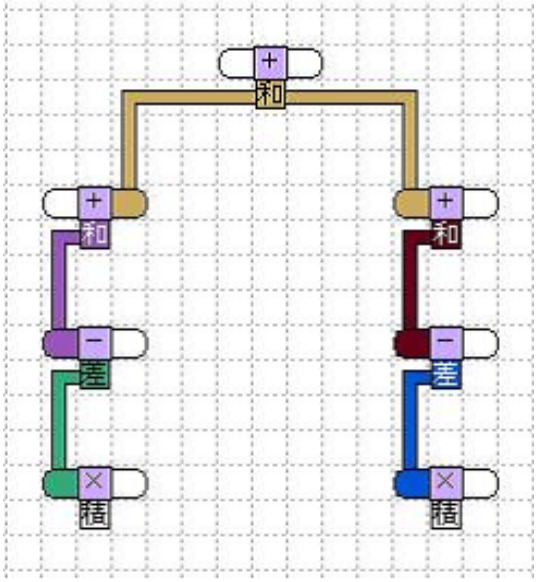

図 3.30: 等間隔である

対称的に配置

一つのパネルに複数のパイプが繋がっている場合は，左右対称にパネルを配置していく．プログラムが横に長くなりキャンバスに収まらなくなるときは，キャンバスサイズの変更を自動で行う．

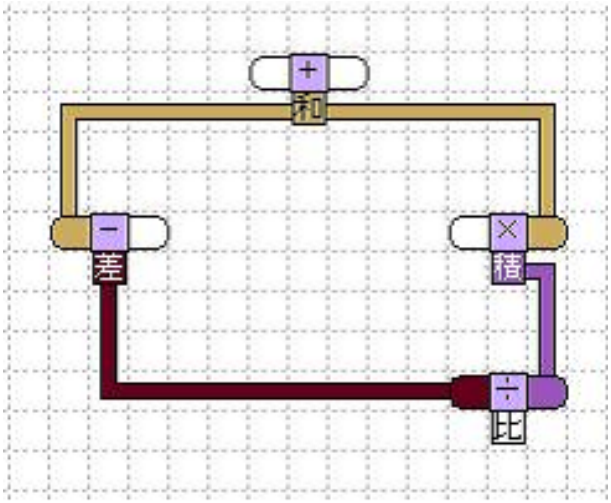

図 3.31: 対称でない

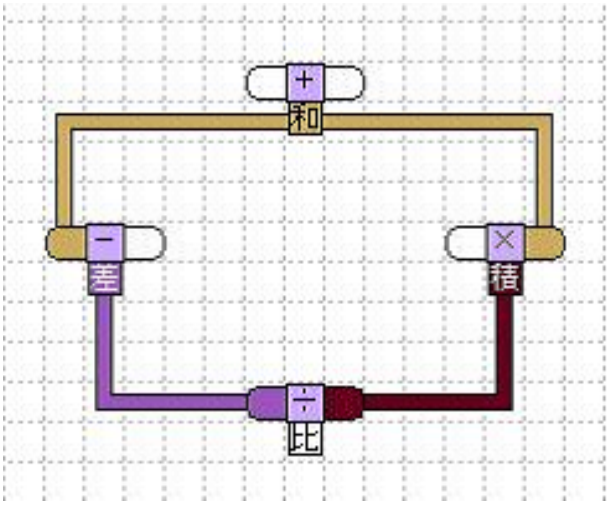

図 3.32: 対称である

3.3 実装

ゆばを用いてプログラム清書システムを実装する．

3.3.1 清書ボタン

ゆばのキャンバス上に清書ボタンというものを設置した．図 3.33 の「清書」という文字が書かれたものが清書ボタンである．

プログラムを記述した後で清書ボタンをクリックすると，そのキャンバス上のプログラムが自動的に清書されるという仕組みである．

図 3.33: 清書ボタン

3.3.2 処理内容

プログラム清書の処理は二つの段階をもって行われる．最初は，とりあえず木構造に近い形になるように配置をする．その後，美しいグラフに改善できる方法がある場合はより良い形に配置し直していく．

1. 木構造に沿った形に整理して配置する

   木構造において，キャンバス上の結果端子を根ノードに，パネルおよびキャンバス上の引数端子を内部ノードまたは葉ノードに見立てて配置していく．このときいくつか規則がある．

   - 各パネル間は等間隔に配置する

     キャンバス大きさと木の深さをから配置する場所を計算する．このとき，必ず一マス分はスペースを作るようにする．

- 必ず逆方向進むところがないようにする
  複数のパイプでつながっているパネルはより深くなる方に配置する.
- 繋がっているパネルを左右対称に配置する
  基のパネルの位置を軸として，左右のパネル数が同数になるように等間隔で配置する．繋がっているパネルの数が奇数の場合は基のパネルの真上に一つを配置して，残りを左右同数になるようにする.

これにより，ひとまず全体的な構造を形成する.

2. より美しいグラフになるように配置し直す

   - パイプの交差がある
     交差が最も少なくなるパターンを見つける.
   - 二回以上曲がりがあるパイプがある
     曲がりの総数が最も少なくなるパターンを見つける．ゆばでは入力端子と出力端子が一直線上でない場合，必ず一回の曲がりが起こるため二回からチェックを行う.

   これらのパターンが存在する場合は，パネルを配置する順番を変える, パネルの左右を反転させる, パイプを引く順番を変えるといった方法で最良のパターンを見つけ出す．このとき，パイプの交差を減らすことを優先するようにした．最も交差の総数が少なくなる場合を決め，その中で最もパイプの曲がりの総数が少なくなる場合を選び出す.

こうして最終的に選ばれたパターンにプログラムを書き換えて清書の完了である．全体の形としては木構造に近い形で整理されている．また，交差が少ない, 曲がりが少ない, 一方向に進む, 等間隔に配置, 対称的に配置という美しいグラフの基準もすべて満たしている．第 2 章で提案したプログラム清書システムの条件を満たしたものを構築することができた.

3.4 実行例

本研究で構築されたプログラム清書システムを実際にプログラムに使用して，その挙動を示す.

3.4.1　3つの数の平均を求めるプログラム

3.1.4で用いた3つの数の平均を求めるプログラムを清書システムを用いて書き換えた.

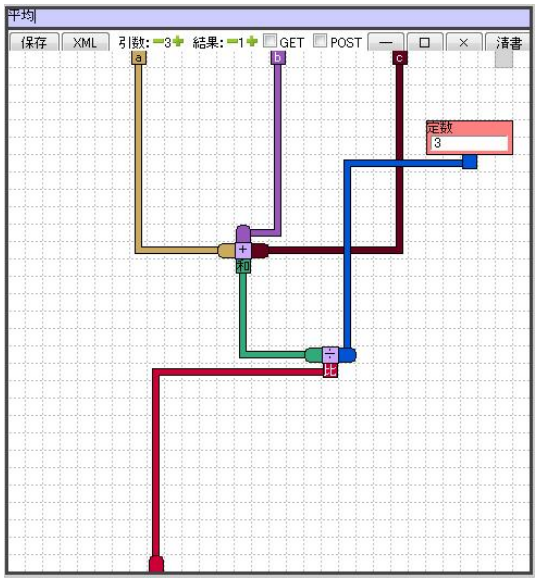

図 3.34: 3つの数の平均を求めるプログラム清書前

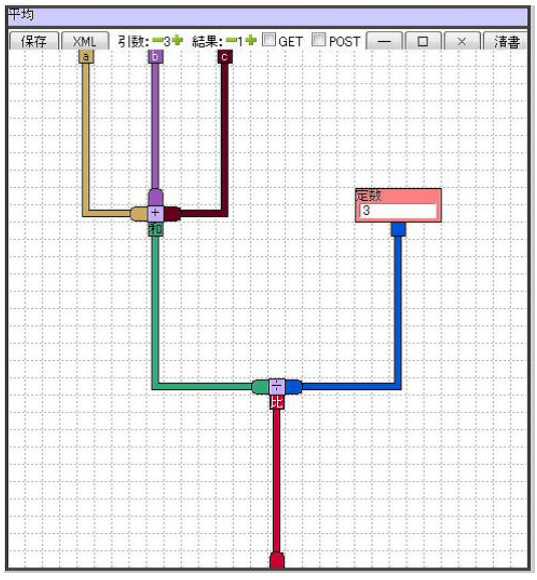

図 3.35: 3つの数の平均を求めるプログラム清書後

簡単なプログラムなのでそれほど大きな変化はないが，交差はなくなり対称的なプログラムに書き換えることができた.

3.4.2　ユークリッドの互除法のプログラム

ユークリッドの互除法のプログラムを清書システムを用いて書き換えた．本来ゆばでは，このような複雑なプログラムを記述することは想定していない．しかしここでは，煩雑なプログラムでも可読性の高いものに書き換えられると言う例を示したかったので，敢えてこのプログラムを使用した．

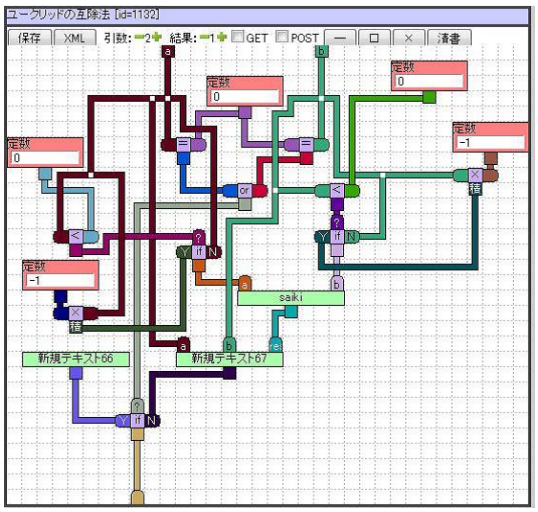

図 3.36: ユークリッドの互除法のプログラム清書前

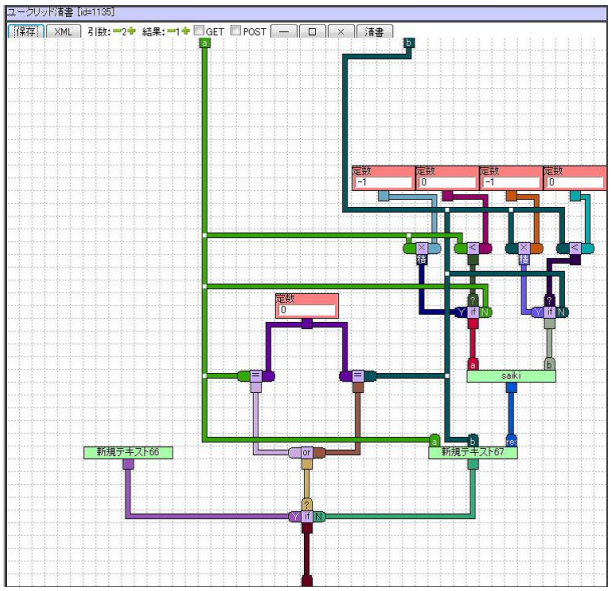

図 3.37: ユークリッドの互除法のプログラム清書後

プログラム清書システム適用前は煩雑で内容を理解することが困難なプログラムであった．しかし，プログラム清書システムを適用後は整理されて読みやすい形になっていると言える．さすがにこのプログラムでは，パネルの交差や曲がりの数は多くなってしまっているが，十分可読性を高めることはできているはずである．

3.4.3 結果

どちらのプログラムもきれいな形に書き換えることができた．特に3.4.2のような煩雑なプログラムの場合は，明らかに理解しやすい形になっている．可読性が高いものに書き換えることができたと言える．

しかし，きれいな形というのは個人の感覚的なものである．このシステムが本当に清書と呼べるものなのか，次の第4章で検証を行う．

第4章　評価

第3章で設計したプログラム清書システムが有用なものであるかを検証し，評価を行う．

4.1　評価方法

ゆばを用いてwebアプリケーションを作成する．そのwebアプリケーションのプログラムを，実際に本研究で構築したプログラム清書システムを用いて清書する．そして，プログラムの清書前と清書後を比較していく．

4.1.1　評価基準

美しいグラフに必要な項目をそれぞれ数値で表して比較する．評価基準となるのは次の5つのポイントである．

交差が少ない

交差の数が少ないほど良い．

曲がりが少ない

曲がりの数が少ないほど良い．

一方向に進む

逆方向に進んでいる数が少ないほど良い．

等間隔に配置

要素間距離の最大値と最小値の差を計算する．差が小さいほど良い．
要素間距離の測り方は，要素間の縦方向のマス目の数を用いる．ただし，距離を測る対照となるのは繋がっている要素の中で最も近いものとする．

対称的に配置

キャンバス上の左右の要素数の差を計算する．差が小さいほど良い．キャンバス中央に縦方向に分割線を引き，それを挟んで右側と左側の要素数を求める．分割線に接している要素は真ん中にあると捉え，どちらにも含まないものとする．

4.2 web 掲示板を用いた評価

ゆばを用いて web 掲示板を作成する．この web 掲示板を 4.1.1 で決めた評価基準に従って比較を行う．

今回評価に利用した web 掲示板は，参考文献 [1] で紹介されているプログラムを，パネルの配置等を含めて全て再現して作成した．著者が自ら作成したプログラムでは，作為的に煩雑なプログラムを作成してしまい正しい評価が行われない恐れがあるからだ．第三者が作成したプログラムを使用することにより正当な評価が行われるはずである．

web 掲示板のプログラム

今回使用する web 掲示板のプログラムの実行結果は図 4.1 のようになっている．この web 掲示板のプログラムの中でメインとなる，記事を 10 件ごとに切り出して表示するプログラムと投稿フォームから投稿された記事を記憶スロットに保存するプログラムの 2 つを利用して比較を行う．

記事を 10 件ごとに切り出して表示するプログラムの清書前を図 4.2 に，清書後を図 4.3 に示す．

投稿フォームから投稿された記事を記憶スロットに保存するプログラムの清書前を図 4.4 に，清書後を図 4.5 に示す．

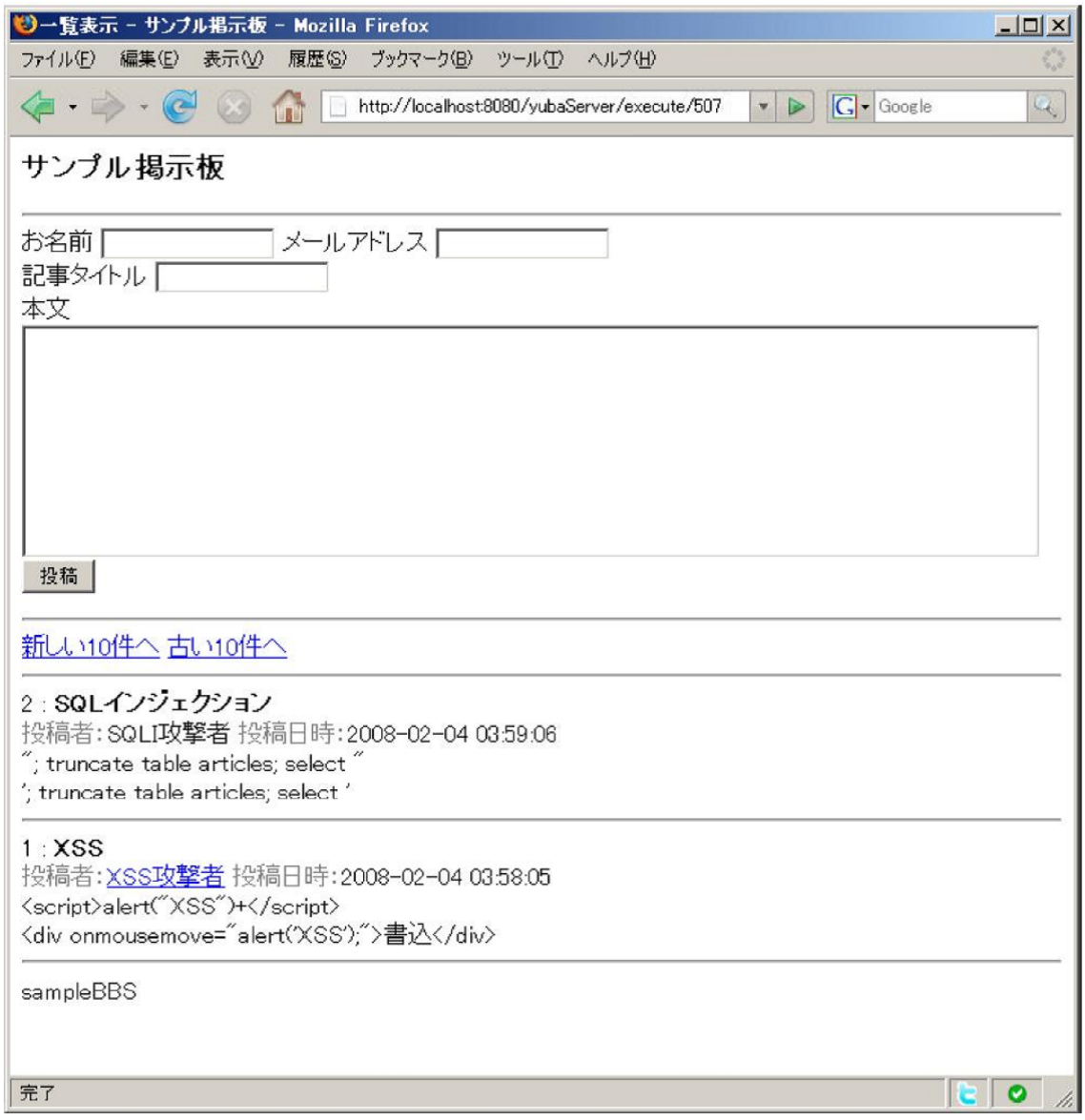

図 4.1: web 掲示板のプログラムの実行結果

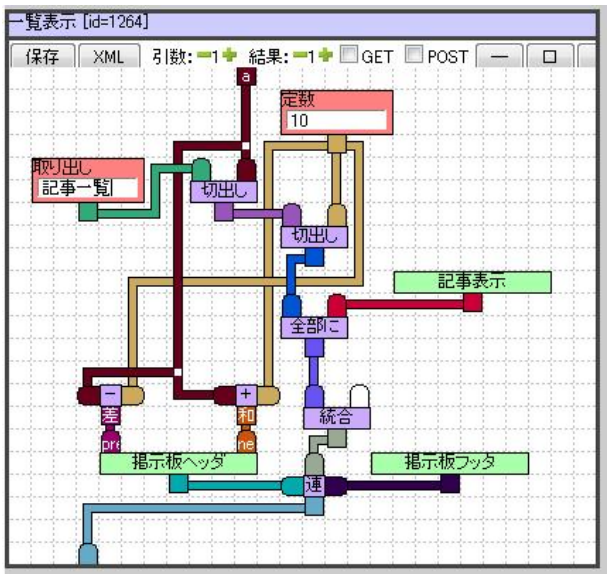

図 4.2: 記事を 10 件ごとに切り出して表示するプログラムの清書前

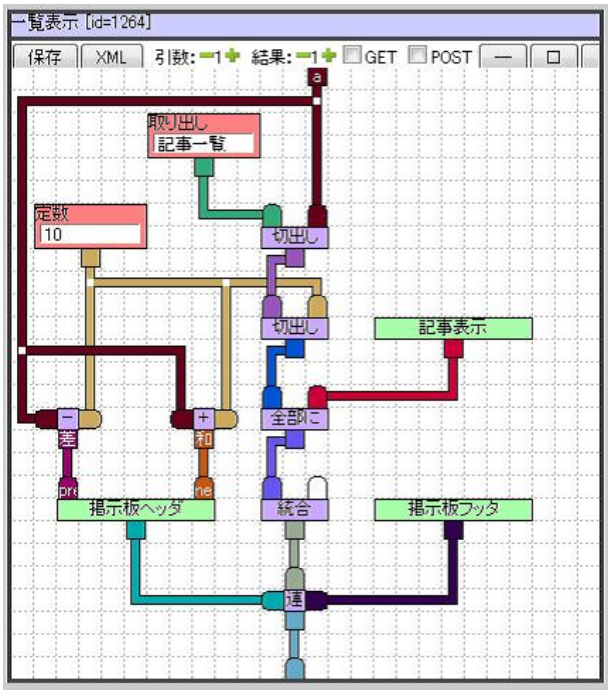

図 4.3: 記事を 10 件ごとに切り出して表示するプログラムの清書後

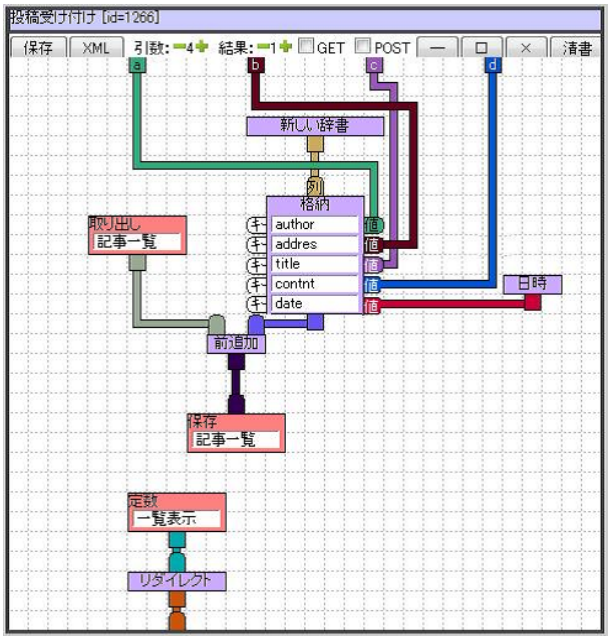

図 4.4: 投稿フォームから投稿された記事を記憶スロットに保存するプログラムの清書前

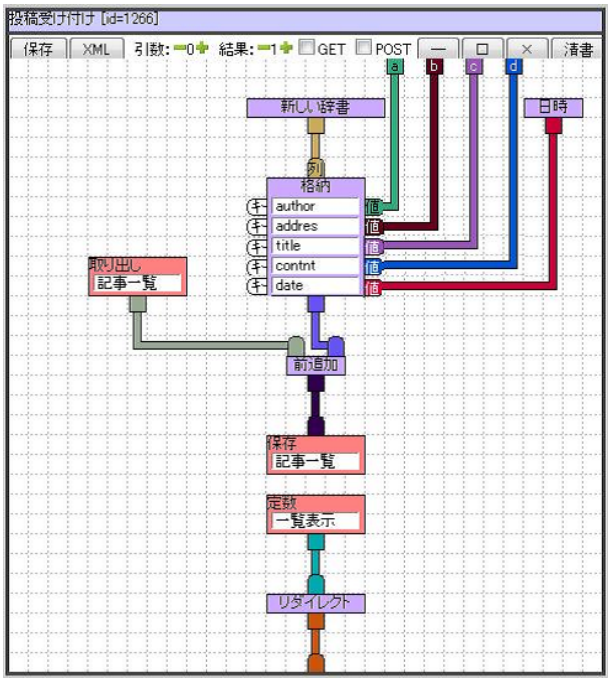

図 4.5: 投稿フォームから投稿された記事を記憶スロットに保存するプログラムの清書後

4.3　比較

記事を 10 件ごとに切り出して表示するプログラムと投稿フォームから投稿された記事を記憶スロットに保存するプログラムを，それぞれ評価基準に沿って比較を行う．また，この 2 つのプログラムの合計値を求め，web 掲示板のプログラム全体としての比較も行う．

4.3.1　記事を 10 件ごとに切り出して表示するプログラムの比較

交差数

清書システム適用前は 6 箇所，適用後は 2 箇所で交差が起こっている．

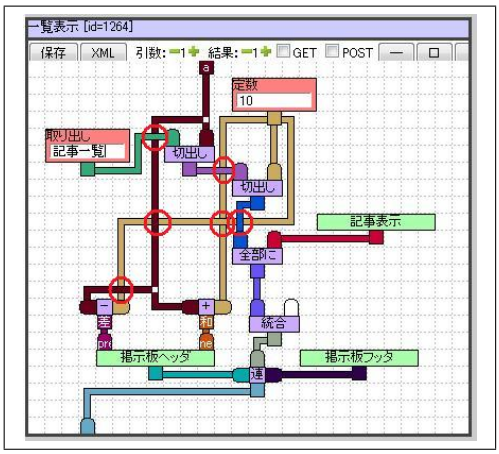

図 4.6: 清書前の交差数

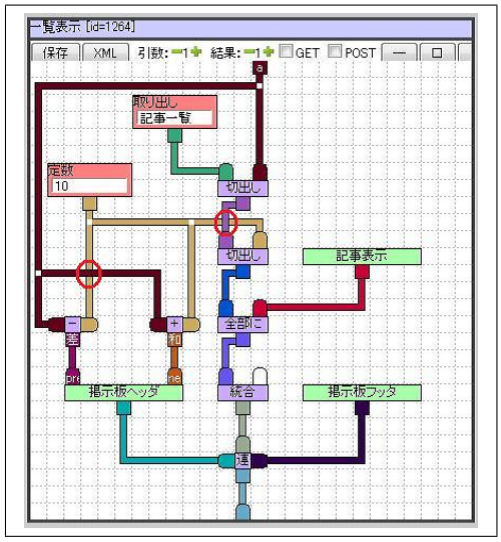

図 4.7: 清書後の交差数

曲がり数

清書システム適用前は 14 箇所，適用後も 14 箇所で曲がりが起こっている．

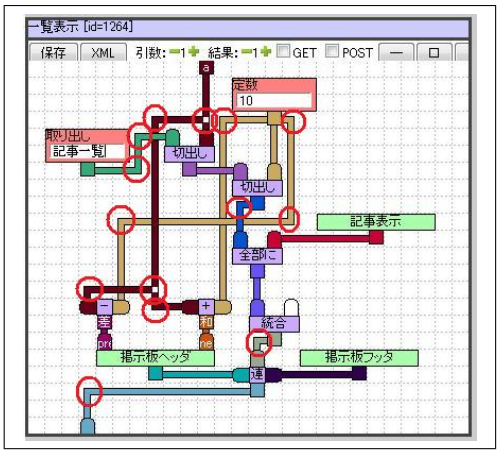

図 4.8: 清書前の曲がり数

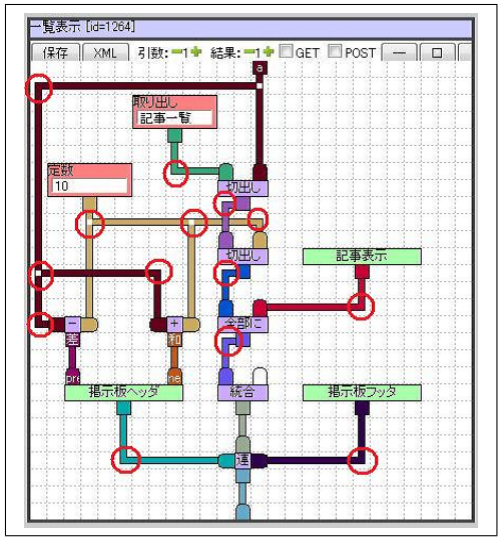

図 4.9: 清書後の曲がり数

逆方向数

清書システム適用前は 1 箇所，逆方向に進むところがある．適用後では逆方向に進むところはない．

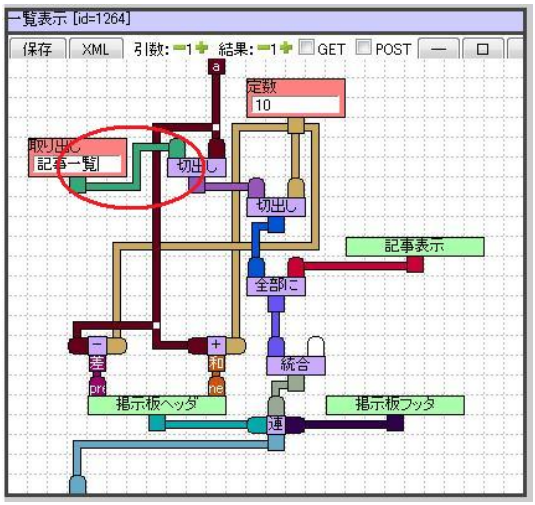

図 4.10: 清書前の逆方向数

要素間差

清書システム適用前の要素間距離の最大値は 3, 最小値は 0 なので，要素間差は 3 となる．適用後では，要素間距離は常に 1 なので，要素間差は 0 となる．

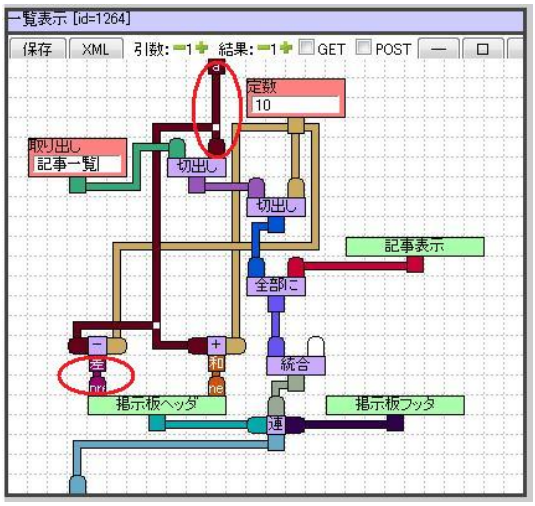

図 4.11: 清書前の要素間距離の最大値と最小値

左右の要素差

清書システム適用前の左側の要素数は 7，右側の要素数は 2 なので，左右の要素差は 5 となる．適用後では，左側の要素数は 5，右側の要素数は 2 なので，左右の要素差は 3 となる．

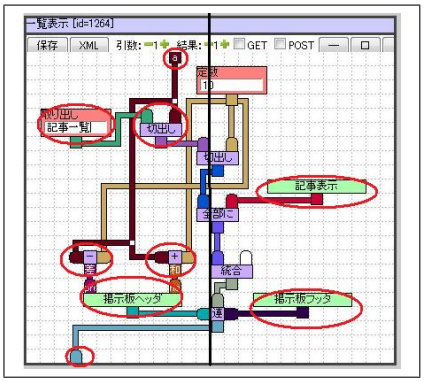

図 4.12: 清書前の左右の要素数

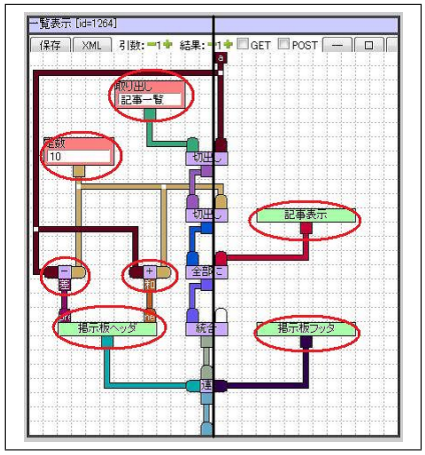

図 4.13: 清書後の左右の要素数

比較表

評価基準の各項目を表 4.1 にまとめた.

表 4.1: 記事を 10 件ごとに切り出して表示するプログラムの比較表

	適用前	適用後
交差数	6	2
曲がり数	14	14
逆方向数	1	0
要素間差	3	0
左右の要素差	5	3

4.3.2 投稿フォームから投稿された記事を記憶スロットに保存するプログラムの比較

交差数

清書システム適用前は 3 箇所で交差が起こっている．適用後では交差は起こっていない.

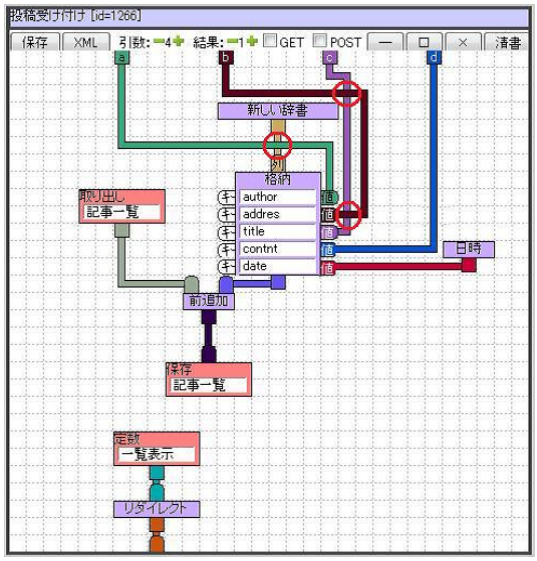

図 4.14: 清書前の交差数

曲がり数

清書システム適用前は 10 箇所，適用後では 7 箇所で曲がりが起こっている．

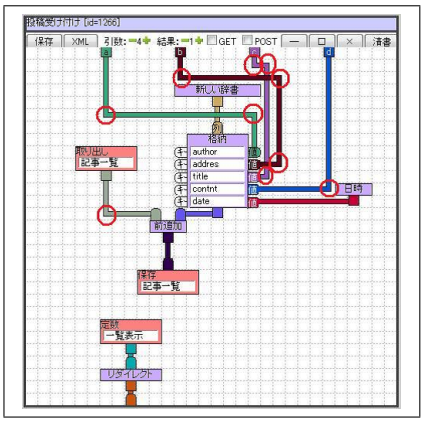

図 4.15: 清書前の曲がり数

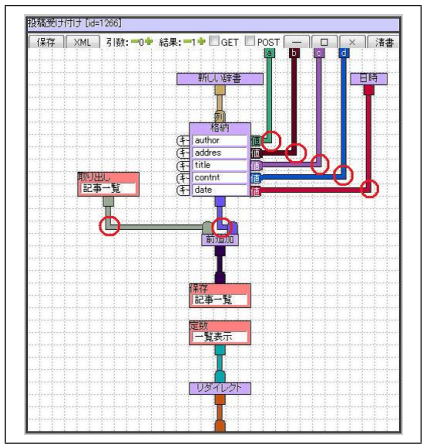

図 4.16: 清書後の曲がり数

逆方向数

このプログラムでは，清書システム適用前，適用後ともに逆方向に進むところはない．

要素間差

清書システム適用前の要素間距離の最大値は 2, 最小値は 0 なので，要素間差は 2 となる．適用後では，要素間距離は常に 1 なので，要素間差は 0 となる．

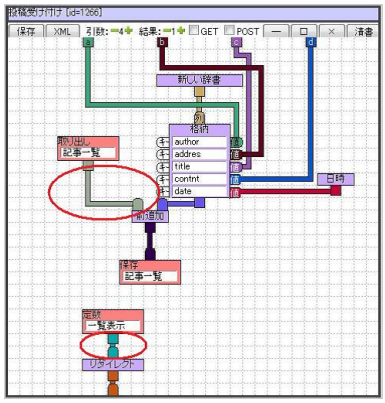

図 4.17: 清書前の要素間距離の最大値と最小値

左右の要素差

清書システム適用前の左側の要素数は 8，右側の要素数は 3 なので，左右の要素差は 5 となる．適用後では，左側の要素数は 1，右側の要素数は 5 なので，左右の要素差は 4 となる．

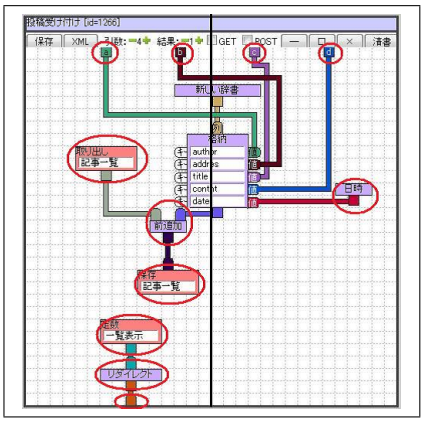

図 4.18: 清書前の左右の要素数

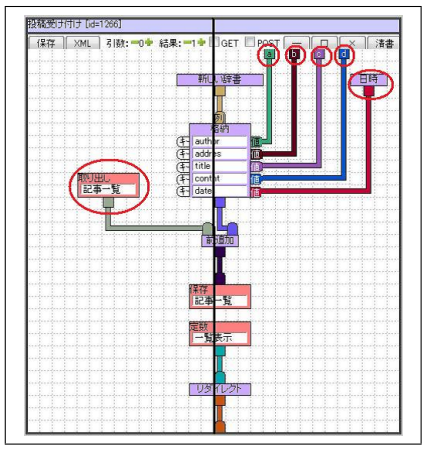

図 4.19: 清書後の左右の要素数

比較表

評価基準の各項目を表 4.2 にまとめた．

表 4.2: 投稿フォームから投稿された記事を記憶スロットに保存するプログラムの比較表

	適用前	適用後
交差数	3	0
曲がり数	10	7
逆方向数	0	0
要素間差	2	0
左右の要素差	5	4

4.3.3 合計値の比較

記事を 10 件ごとに切り出して表示するプログラムと投稿フォームから投稿された記事を記憶スロットに保存するプログラムの評価基準を，各項目毎に合計した値を表 4.3 にまとめた．

表 4.3: 2 つのプログラム合計値の比較表

	適用前	適用後
交差数	9	2
曲がり数	24	21
逆方向数	1	0
要素間差	5	0
左右の要素差	10	7

4.4 評価結果

僅差の項目もあるが，全ての評価基準において清書システム適用後のほうが良い値となった．これは明らかに可読性の高いプログラムに書き換えることができたと言える．

今回利用した web 掲示板のプログラムは論文に掲載されていたものである．そのため，ある程度きれいなプログラムを書くように意識していたと考えられる．その結果，期待していたほど大きな数値の差にはならなかった．それでも全ての項目で清書システム適用後のほうが優れた結果となった．よって，このプログラム清書システムは有用であると言える．

第5章　まとめ

本論文の全体のまとめを行う．本研究で構築したプログラム清書システムの考察をした後に，今後の課題と結論を述べる．

5.1　考察

美しいグラフの基準である交差が少ない，曲がりが少ない，一方向に進む，等間隔に配置，対称的に配置という，5つ全ての条件を満たすプログラムに書き換えるシステムを構築することができた．第4章の評価によって，プログラム清書システムを適用することによって可読性の高いプログラムに書き換えられることが証明された．よって，本研究で構築したシステムは清書システムとして成り立っていると言える．

5.2　今後の課題

本論文で定めだプログラム清書の基準の中で，対称性だけは著しく低下してしまう場合がある．一つの要素に繋がっている要素同士を左右対称に並べることで対称的なプログラムになるとした．しかし，繋がった先の要素の量までは考えられていない．そのため，極端に要素の量に差があるものが繋がっていた場合に，左右のバランスが悪くなってしまい対称性が低いプログラムとなってしまう．それぞれ繋がっている要素の量を求め，それを基に重み付けを行うことでより対称性を高くすることができるようになるはずだ．

また，プログラムを簡潔に記述し直すシステムも構築できればよかった．不要なものは排除し，まとめられるものはまとめることで，プログラム要素自体を減らすことができる．これによりすっきりとした簡潔なプログラムに書き換えることができる．今回プログラム清書システムを実装したゆばには，3.1.2で示したようにプログラムを部品化する機能がある．この機能を利用して，複数のパネルを一つのパネルにまとめていくシステムを作ることができれば，より優れたプログラム清書システムになっていたはずだ．

5.3 結論

ビジュアルプログラミング言語を清書するシステムを構築することができた．このシステムを利用することにより，煩雑なプログラムでも読みやすく整理された形に書き換えることができる．ビジュアルプログラミング言語の大きな特徴である可読性の高さを更に高めることができる．これにより，プログラミング初心者でも内容を理解しやすくなり，web サービスを開発するための敷居を下げることができる．開発に関わったことがない人でも，web サービスを利用する中で改善したいポイントやあれば便利なサービスを思いつくはずだ．このプログラム清書システムを用いた開発環境があれば，今までは実現できずに埋もれていたアイディアの発信を手助けすることができる．

本研究で構築されたプログラム清書システムを利用して，新しい web サービスが世の中に発信されることを期待したい．

謝辞

本研究を終えるにあたり，ご多忙にもかかわらず終始適切なご指導をいただきました，筧捷彦教授に心から感謝します.

参考文献

[1] 三浦琢磨: ウェブサイト構築を目的としたデータフロー型ビジュアルプログラミング言語「ゆば」の開発, 2007 年度早稲田大学大学院理工学研究科情報ネットワーク専攻修士論文, 2008.

[2] 佐藤幸弘: ビジュアルプログラミングを用いた web サービス開発環境の開発, 2008 年度早稲田大学大学院理工学研究科情報ネットワーク専攻修士論文, 2009.

[3] E. クライツィグ 著, 多村義保 訳: 最適化とグラフ理論, 培風館, 2003.

[4] 落合豊行: グラフ理論入門—平面グラフへの応用, 日本評論社, 2004.

[5] 増井俊之: 進化的学習機構を用いたグラフ配置制約の自動抽出, インタラクティブシステムとソフトウェア II: 日本ソフトウェア科学会 WISS'94 pp195-204, 近代科学社, 1994.

[6] 西尾元宏: 板ばねモデルを用いたインタラクティブな曲線グラフ描画手法とその応用に関する研究, 東京大学大学院工学系研究科環境海洋工学専攻修士論文, 2003.

[7] 梶永泰正, 伊藤貴之, 池端裕子, 土井淳, 井上恵介: 力学モデルを用いたウェブサイトの視覚化, 可視化情報学会第 29 回可視化情報シンポジウム論文集, 2001.

[8] 久保田秀和: 絵画的プログラミング，人工知能学会主催 第 9 回 AI 若手の集い（MYCOM2009）オンライン予稿集発表番号 (5-3)，2009.

[9] 増井俊之: インタフェースの街角 (10) ビジュアル・プログラミング, UNIX MAGAZIN 1998.9, 1998.

[10] 田中二郎: 「ビジュアルプログラミング」第 2.3 章, bit 別冊ビジュアルインタフェース pp65-78, 共立出版, 1996.

[11] 杉原厚吉: 清書プログラム, 情報処理 Vol.20 No.11 pp970-973, オーム社, 1979.